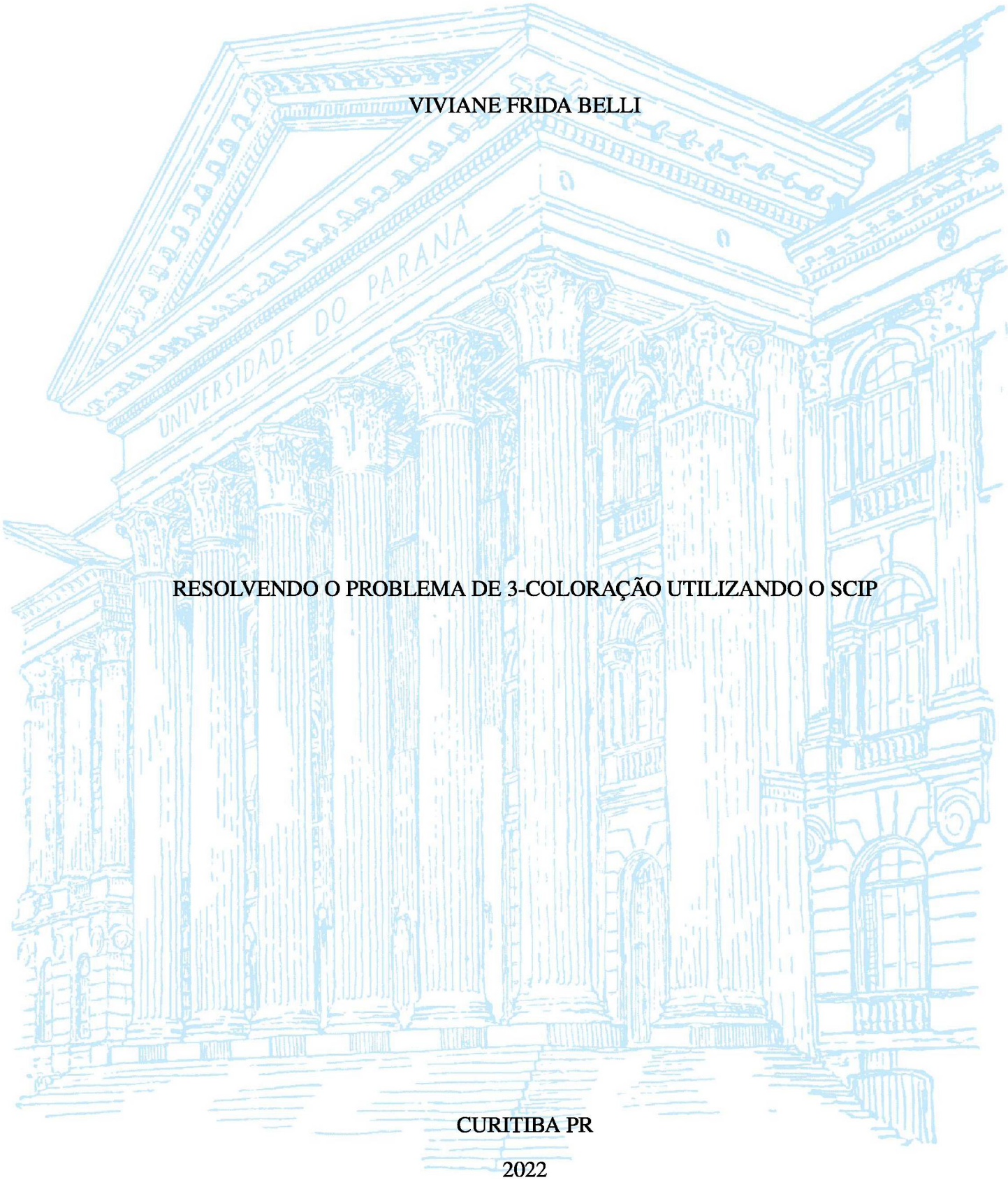
UNIVERSIDADE FEDERAL DO PARANÁ

VIVIANE FRIDA BELLI



RESOLVENDO O PROBLEMA DE 3-COLORAÇÃO UTILIZANDO O SCIP

CURITIBA PR

2022

VIVIANE FRIDA BELLI

RESOLVENDO O PROBLEMA DE 3-COLORAÇÃO UTILIZANDO O SCIP

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André Luiz Pires Guedes.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Belli, Viviane Frida
Resolvendo o problema de 3-coloração utilizando o SCIP / Viviane Frida Belli. – Curitiba, 2022.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.

Orientador: André Luiz Pires Guedes

1. Otimização combinatória. 2. Grafos. 3. Solving Constraint Integer Programs. 4. Satisfabilidade Booleana. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Informática. III. Guedes, André Luiz Pires. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **VIVIANE FRIDA BELLI** intitulada: **Resolvendo o problema de 3-coloração utilizando o SCIP**, sob orientação do Prof. Dr. ANDRÉ LUIZ PIRES GUEDES, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestra está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 12 de Dezembro de 2022.

Assinatura Eletrônica
13/12/2022 14:59:59.0
ANDRÉ LUIZ PIRES GUEDES
Presidente da Banca Examinadora

Assinatura Eletrônica
14/12/2022 15:30:34.0
KARINA GIRARDI ROGGIA
Avaliador Externo (UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC)

Assinatura Eletrônica
13/12/2022 14:45:11.0
FABIANO SILVA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Dedico esta conquista à Viviane de 20 anos atrás.

AGRADECIMENTOS

Esta dissertação é resultado de muito esforço e só foi possível graças a uma rede de apoio incrível. Além de ser um trabalho desafiador por si só, foi toda desenvolvida enquanto vivemos a pandemia de COVID-19. Vivemos momentos de incertezas, medo e isolamento, mas ao final sentimos reacender a esperança e a vontade de seguir em frente.

Agradeço ao meu orientador, professor André Luiz Pires Guedes, que além de guiar todo o desenvolvimento deste trabalho foi também acolhimento em momentos difíceis deste processo.

Agradeço a todos os professores do DINF, por conseguirem rapidamente adaptar suas aulas e mesmo remotamente estarem presentes da melhor maneira possível. Sem seus esforços e disponibilidade esta conclusão não seria possível.

Agradeço ao meu amado filho Johnny, minha motivação diária para ser sempre uma pessoa melhor.

Agradeço aos familiares, principalmente minha irmã Zinha e minha sogra Maria, por suas orações e torcida.

Agradeço às minhas amigas, mulheres tão diferentes e todas tão inspiradoras!

Obrigada também ao Instituto Federal Catarinense, meu local de trabalho e incentivador do desenvolvimento das pessoas e da Ciência.

Obrigada à Universidade Federal do Paraná, por me permitir continuar acreditando na educação pública, gratuita e de qualidade!

E meu maior agradecimento ao meu melhor amigo, e por uma grande sorte na minha vida, meu marido Emerson Ruthes. Sem seu apoio, seus cafés, sua alegria e incentivo, eu não teria conseguido. Seguimos juntos, estudando, rindo, crescendo.

RESUMO

Neste trabalho propomos uma rotina de pré-processamento a ser executada em instâncias do problema de 3-coloração de grafos no intuito de reduzir o tempo necessário para sua resolução com o SCIP (*Solving Constraint Integer Programs*), que é uma ferramenta para resolução de problemas de otimização de diferentes categorias. Esta rotina de pré-processamento proposta é executada em instâncias enquanto grafos, sendo esta apenas uma das possíveis estratégias para a redução do tempo necessário para a resolução de problemas de otimização. Com a finalidade de verificação da eficiência, criamos instâncias 3-coloríveis com diferentes níveis de aplicabilidade desta rotina de pré-processamento, considerando a modelagem do problema de 3-coloração de grafos como SAT (Satisfabilidade Booleana).

Palavras-chave: Otimização. SCIP. 3-Coloração. SAT.

ABSTRACT

In this work we propose a pre-processing routine to be executed in instances of 3-coloring graphs problems in the intuitive way of reducing the time required for its resolution with SCIP (Solving Constraint Integer Programs), which is a tool for solving optimization problems of different categories. This proposed pre-processing routine runs on instances as graphs, which is just one of the possible strategies to reduce the time needed to solve optimization problems. In order to verify the efficiency, we created 3-colorable instances with different levels of applicability of this pre-processing routine, considering the modeling of the problem of 3-coloring of graphs as SAT (Boolean Satisfiability).

Keywords: Optimization. SCIP. 3-Coloring. SAT.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A otimização pode ser entendida como um conceito subjacente na análise de problemas complexos de tomada de decisão ou de alocação de recursos [Luenberger et al., 1984]. Pensando em otimização, a decisão a ser tomada sobre um problema complexo envolve a seleção de valores para um certo número de variáveis que mantêm alguma relação entre si. Esta seleção de valores tem o propósito de atender um determinado objetivo, seja quantificar a performance de uma operação ou mensurar a qualidade de uma decisão. Este objetivo de quantificar a maior ou menor performance, mensurar a melhor ou pior decisão, poderá levar em consideração restrições que irão limitar a seleção dos valores para as variáveis envolvidas. O objetivo da otimização será então, de maximizar ou minimizar os resultados, respeitando as possíveis limitações da situação.

Ainda, de acordo com Biegler, a maioria das coisas podem ser melhoradas. Então, no contexto de processos, cientistas e engenheiros otimizam [Biegler, 2010].

Neste contexto, a Matemática e a Ciência da Computação abordam os problemas de otimização. O objetivo destes problemas é encontrar os valores extremos de uma função, ou seja, o maior ou o menor valor possível desta função, dentro um conjunto de valores possíveis.

Entre os valores possíveis para a função de um problema de otimização, todos podem ser chamados de solução do problema, porém apenas os extremos da função, maior ou menor, dependendo do objetivo do problema, podem ser chamados de solução ótima.

Encontrar esta solução ótima pode ser extremamente difícil, dependendo do tipo de problema. Para isso, existem diferentes métodos e técnicas, desenvolvidos e extensamente aprimorados, que são utilizados por desenvolvedores em suas aplicações ou softwares com o intuito de tornar a resolução dos problemas mais prática e eficiente.

No caso específico de utilização de softwares resolvedores de problemas de otimização, é interessante conhecer a técnica aplicada por este na resolução de determinado tipo de problema, pois a sua eficiência pode influenciar no desempenho geral da aplicação.

A motivação deste trabalho é então, analisar a resolução de um determinado tipo de problema de otimização, utilizando como ferramenta o SCIP.

Demonstraremos como o SCIP resolve problemas de otimização combinatória, quais técnicas utiliza e como é feita a entrada de dados dos problemas, abordando especificamente o problema de 3-Coloração de grafos.

A coloração de grafos é um dos assuntos mais populares na teoria dos grafos, sendo a motivação de novas pesquisas inspirada tanto em interesses puramente teóricos quanto na perspectiva da possibilidade de aplicações práticas [Casselgren, 2011].

Em seguida, propomos um experimento para redução de instâncias desta categoria de problemas antes de serem resolvidos pelo SCIP, e faremos uma análise do tempo da técnica aplicada pelo resolvedor e da eficiência do experimento na resolução final do problema.

O SCIP é a ferramenta que se busca analisar por possibilitar a resolução de diversos tipos de problemas. De acordo com seus desenvolvedores, o SCIP é um dos mais rápidos resolvedores de problemas de programação linear inteira mista e programação não-linear inteira mista. Permite total controle do processo de resolução dos problemas e acesso detalhado a informações de todos os passos deste processo.

Este trabalho está organizado em 6 capítulos.

No capítulo 2, abordamos definições e notações de problemas de otimização em geral, modelos e modelagem. No capítulo 3, tratamos especificadamente do problema de 3-Coloração

de grafos, iniciando com histórico e exemplo de aplicação da coloração de grafos em geral e em seguida detalhamos a modelagem do problema de 3-Coloração como CSP e como SAT.

A ferramenta SCIP, tipos de problemas que resolve, maneira como recebe informações do problema e exemplos de resolução de problemas diversos são apresentados no capítulo 4.

O capítulo 5 é dedicado a expor o experimento realizado, nossa sugestão de rotina de pré-processamento, apresentação das instâncias utilizadas e discussão dos resultados. Encerramos com o capítulo 6, propondo o desenvolvimento de trabalhos futuros.

2 PROBLEMAS DE OTIMIZAÇÃO

Em uma definição prática de problema de otimização: dado um sistema ou um processo, busca-se encontrar a melhor solução para este processo, com algumas restrições. Esta tarefa requer os seguintes elementos: uma função objetivo, um modelo confiável e as variáveis do modelo. A função objetivo é necessária para determinar o desempenho que se busca medir, e que será maximizado ou minimizado, dependendo do modelo. Esta função pode medir o custo de um processo, o lucro, o rendimento, etc. Um modelo confiável é aquele que descreve o comportamento de um sistema ou processo e é obtido através do processo de modelagem. E as variáveis que aparecem neste modelo são aquelas que precisarão ser determinadas e ajustadas para atender as restrições do sistema.

2.1 NOTAÇÃO E DEFINIÇÕES

- **Modelo:** De acordo com Carter et al. [2018], um modelo é uma representação idealizada ou uma simplificação de um objeto real, um processo real ou um sistema real. Tratando-se de modelo matemático nesta definição, utiliza-se estruturas matemáticas como equações, inequações, funções, matrizes e operações para representar os componentes do modelo.

- **Modelagem:** Para que seja possível chegar à solução de um problema de otimização, seja esta a solução viável ou a solução ótima, inicialmente é necessária a criação de um modelo matemático que represente tal problema. Determinar o modelo confiável de um problema de otimização é traduzí-lo em um conjunto de equações e/ou inequações que descrevem o comportamento e objetivos do sistema e representam as suas restrições [Biegler, 2010]. Esta etapa é chamada de modelagem e é extremamente importante, pois, caso o modelo não expresse adequadamente o sistema ou processo que se busca otimizar, a solução obtida pelo método de otimização não será a melhor solução para o problema real.

Genericamente, um **problema de otimização** pode ser definido pelo seguinte modelo [Papadimitriou e Steiglitz, 1998]:

$$\begin{array}{ll} \textbf{minimizar} & f(x) \\ \text{(ou maximizar)} & \\ \textbf{sujeito a:} & x \in V \subseteq U \end{array} \tag{2.1}$$

onde,

$f(x)$: é a função objetivo, a função que irá determinar o critério de escolha do valor que otimiza o problema;

x : é a variável de decisão, $x = (x_1, x_2, x_3, ..., x_n)$;

V : é o conjunto dos valores possíveis, também chamado de conjunto viável (ou domínio). Os valores de V atendem a todas as possíveis restrições do problema;

U : é um conjunto do Universo, pode ser $\mathbb{N}^n$, $\mathbb{Z}^n$, $\mathbb{Q}^n$, $\mathbb{R}^n$, etc., e $V \subseteq U$.

Neste modelo genérico de problema de otimização, resolver o problema é encontrar um valor $v^* \in V$, tal que o valor da função f para v^* seja o menor possível, ou seja,

$$f(v^*) \leq f(v) \forall v \in V$$

Nestas condições, existindo valores $v \in V$, qualquer v pode ser chamado de solução viável, mas somente v^* é chamado de solução ótima do problema.

Não existindo valores $v \in V$, a solução do problema é inviável. Ou seja, não existem valores que atendam às possíveis restrições do problema.

E, se não existir valor $w \in V$ tal que $f(w) \leq f(v)$ (no caso de minimização) para todo $v \in V$, então a solução do problema é ilimitada, o que significa que as restrições que determinam o conjunto V não o limitam.

Esta determinação das soluções do problema é válida com a condição de que $f(v)$, para todo $v \in V$, é um valor finito.

Particularmente, para os problemas de otimização combinatória Papadimitriou e Steiglitz [1998] definem a sua solução como um objeto que é geralmente um número inteiro, um subconjunto, uma permutação, ou a estrutura de um grafo.

2.2 MODELAGEM DOS PROBLEMAS DE OTIMIZAÇÃO

O processo de modelagem de um problema de otimização pode ser definido em etapas, de acordo com Carter et al. [2018].

Este processo se inicia com a definição do problema a ser modelado. Em seguida deve-se realizar a observação do sistema real, determinar quais aspectos do sistema são controláveis e quais não são, identificar os objetivos e propósitos do sistema e as restrições ou limitações que o controlam. Esta coleta de dados significa compreender o objetivo desejado do sistema e expressá-lo em termos matemáticos, fazendo a tradução de um problema real em um problema de otimização. Nesta etapa ocorre a determinação das variáveis do sistema, bem como a formulação das restrições e da função objetivo. O sucesso desta etapa resume todo o processo de modelagem. Após esta etapa, este modelo matemático deverá ser validado através da sua aplicação em cenários reais.

Na aplicação do modelo, pode-se constatar que o modelo seja uma representação imperfeita do sistema real, então o processo de modelagem pode reiniciar, para ser melhor elaborado. Feita a validação do modelo, ele deve ser então implementado para ser utilizado como uma ferramenta no auxílio de tomada de decisão ou de análise da eficiência do sistema. Nas etapas de validação e implementação do modelo deve-se utilizar ferramentas que possibilitem alterações no objetivo e nas restrições do sistema. Com a utilização de linguagens de modelagem e softwares resolvedores é possível realizar as adequações necessárias no modelo. A Figura 2.1 ilustra as etapas deste processo de modelagem.

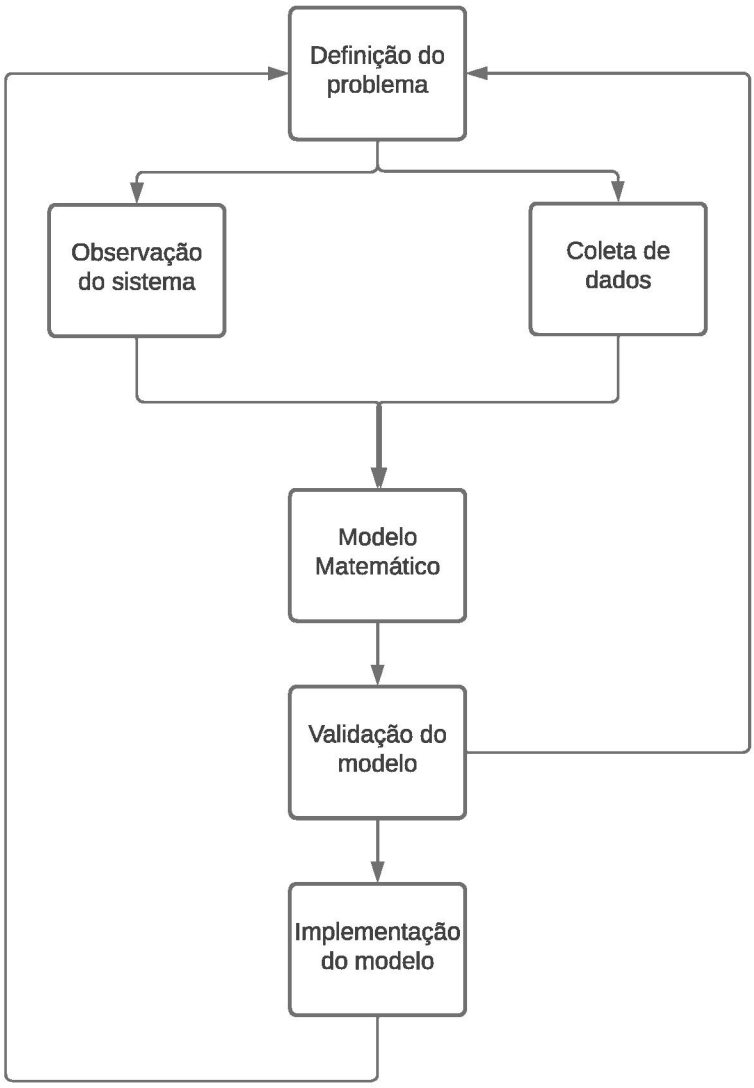


Figura 2.1: Etapas do processo de modelagem

A etapa de definição do modelo matemático de problemas de otimização pode ser orientada pela seguinte sequência de passos:

1. identificar as variáveis do problema: $x_1, x_2, x_3, ..., x_n$;
2. identificar as constantes do problema: $c_1, c_2, c_3, ..., c_m$;
3. identificar os valores dos fatores limitantes do problema: $b_1, b_2, b_3, ..., b_z$;
4. determinar a função objetivo;
5. determinar as restrições (inequações e equações) e não-negatividade das variáveis, quando for o caso;

Abaixo citamos um exemplo de modelagem de um problema clássico:

Problema da mistura de metais [Ravindran, 2016]

Uma empresa siderúrgica possui quatro tipos diferentes de misturas de metais, SM-1, SM-2, SM-3 e SM-4, com suas respectivas composições e custo por unidade de medida, conforme tabela 2.1

A empresa precisa preparar uma mistura destes quatro tipos de metais, de tal maneira que a composição da mistura final tenha a seguinte proporção por peso: Al: 4,43%, Si: 3,22%, C: 3,89% e Fe 88,46%, e ainda, como menor custo possível. Como podem preparar esta mistura?

Tabela 2.1: Composição de cada tipo de metal

	% de cada elemento na composição				Custo por unidade de medida
Metal	Al	Si	C	Fe	R$
SM-1	5	3	4	88	100,00
SM-2	7	6	5	82	150,00
SM-3	2	1	3	94	97,00
SM-4	1	2	1	96	95,00

Para responder esta pergunta, é necessário determinar a proporção de cada um dos quatro tipos de metais (SM-1 a SM-4) na preparação da mistura.

Passo 1: identificação das variáveis do problema:

neste problema, a variável de decisão é x_j, que corresponde à proporção de peso do metal SM-j na mistura final, sendo $j = \{1, 2, 3, 4\}$.

Passos 2 e 3: identificação das constantes e os valores limitantes do problema:

a proporção por peso do elemento Al na mistura final deve ser: $5x_1 + 7x_2 + 2x_3 + 1x_4$ e totalizar 4,43%;

a proporção por peso do elemento Si na mistura final deve ser: $3x_1 + 6x_2 + 1x_3 + 2x_4$ e totalizar 3,22%;

a proporção por peso do elemento C na mistura final deve ser: $4x_1 + 5x_2 + 3x_3 + 1x_4$ e totalizar 3,89%;

a proporção por peso do elemento Fe na mistura final deve ser: $88x_1 + 82_2 + 94_3 + 96x_4$ e totalizar 88,46%;

e a mistura final será uma soma das proporções de cada metal que a compõem:

$x_1 + x_2 + x_3 + x_4 = 1$

Passo 4: determinar a função objetivo:

Neste problema, o objetivo é obter a mistura final com o menor custo total possível, considerando o custo por unidade de medida de cada metal e que tenha em sua composição quantidades x_j de cada metal:

minimizar $100x_1 + 150x_2 + 97x_3 + 95x_4$

Passo 5: determinar as restrições do problema:

de acordo com a quantidade de cada metal que deverá estar presente na mistura, temos o seguinte sistema de equações que representam as restrições do problema:

$$x_1 + x_2 + x_3 + x_4 = 1$$
$$5x_1 + 7x_2 + 2x_3 + 1x_4 = 4,43$$
$$3x_1 + 6x_2 + 1x_3 + 2x_4 = 3,22$$
$$4x_1 + 5x_2 + 3x_3 + 1x_4 = 3,89$$
$$88x_1 + 82x_2 + 94x_3 + 96x_4 = 88,46$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

$$x_3 \geq 0$$
$$x_4 \geq 0$$

Saber modelar corretamente um problema é muito importante, pois definido o seu modelo, é possível classificar este problema de acordo com suas características. Desta maneira, sabendo-se identificar o tipo de problema que se deseja otimizar, é possível determinar o seu método de resolução, pois estes métodos são, de forma geral, específicos para cada tipo de problema.

Esta determinação do tipo do problema é feita com base na identificação de suas seguintes características:

- tipo das variáveis: se discretas, contínuas ou ambas;
- características da função-objetivo e das funções de restrição (por exemplo: linearidade);
- presença de restrições;
- critérios da função objetivo;

Durante o desenvolvimento de uma aplicação, sabendo o tipo do problema a ser resolvido, é possível a utilização de algoritmos eficientes, através de softwares resolvedores. Para o desenvolvedor, esta definição é bastante conveniente, bem como saber de que maneira a ferramenta que está utilizando recebe os dados de entrada da função objetivo e das restrições do problema. [Matousek e Gärtner, 2007]

3 O PROBLEMA DE 3-COLORAÇÃO

O problema de coloração, e por conseguinte o problema de 3-coloração de grafos, está na categoria dos problemas de otimização discretos, chamada de Otimização Combinatória. É um problema da classe NP-Completo amplamente estudado, com abordagens diversas. Um completo levantamento bibliográfico de maneiras de solucionar este problema, análises e aplicações foi desenvolvido por Chiarandini e Gualandi [2015].

Definição: Sendo $G = (V, E)$, um grafo a ser colorido, onde V e E correspondem aos conjuntos dos vértices e arestas respectivamente, a **k-coloração** do grafo corresponde a uma atribuição de cores (ou valores) aos vértices do grafo, de um conjunto com k cores, de forma que considerando qualquer aresta $\{i, j\} \in E$, as cores de i e de j são distintas.
Resposta do Problema de k-Coloração: SIM, se for possível colorir G com até k cores; NÃO, caso contrário. [Lima, 2016]

3.1 HISTÓRICO

A origem do estudo do problema de coloração de grafos pode ser atribuída a Francis Guthrie (1831-1899), matemático que no século XIX conjecturou ser possível colorir qualquer mapa de países com um número máximo de quatro cores, de maneira que os países com fronteiras em comum sejam coloridos com cores diferentes, como o exemplo da Figura 3.1. Apesar de não ter conseguido uma demonstração matemática, sua conjectura interessou também o professor Augustus de Morgan (1806-1871), que foi o responsável pela divulgação do problema entre seus alunos e colegas. Desta maneira, no ano de 1878 foi publicado o artigo *The solution of a problem which recently achieved some renown*, no periódico *Nature*. Com esta divulgação, o problema levantou o interesse também de Alfred Bray Kempe (1821-1895), que publicou uma demonstração de que é possível colorir qualquer mapa com quatro cores, em 1879. Porém, esta demonstração foi refutada por Percy John Heawood (1861-1955) em 1890, ao encontrar um erro na demonstração de Kempe e demonstrar o Teorema das Cinco Cores [Lima, 2016].

Figura 3.1: Mapa dos Condados Cerimoniais da Inglaterra - Colorido com 4 cores.
Fonte: https://pt.dreamstime.com/mapa-dos-condados-cerimoniais-da-inglaterra-de-cores-etiquetadas-do-pais-europeu-image177566809. Acesso em 07/07/2022.

Após esta refutação, alguns estudos conseguiram demonstrar a validade do Teorema das Quatro Cores, porém para mapas com número limitado de faces (ou países), conforme tabela 3.1.

Ano	Autor	Faces
1920	Philip Franklin	25
1926	C.N. Reynolds	27
1936	Philip Franklin	31
1938	C.E. Winn	35
1968	Oystein Ore e Joel Stemple	40

Tabela 3.1: Demonstrações do Teorema das Quatro Cores, adaptado de Sousa [2001]

Apenas em 1976, com a ajuda de um IBM 360 e mais de mil horas de processamento, conseguiu-se provar esta conjectura sendo então enunciado o Teorema das Quatro Cores por Appel et al. [1977].

Esta demonstração, apesar de polêmica, continua sendo aceita, mas o seu desenvolvimento sem a necessidade de computadores continua em aberto.

Com os estudos sobre o Teorema das Quatro Cores, muito se desenvolveu na literatura do problema de coloração de vértices e determinação do número cromático de grafos.

De maneira geral, a coloração de um grafo consiste na atribuição de cores aos seus vértices, arestas, faces de um grafo planar, ou uma combinação destes simultaneamente [Kubale, 2004].

Definição: O **problema de coloração dos vértices** compreende a determinação de uma cor para cada vértice do grafo, de maneira que as cores de vértices adjacentes sejam diferentes, utilizando um número mínimo de cores.

De acordo com Garey e Johnson [1979], a coloração de vértices é um problema da classe NP-Difícil e são diversas as aplicações do problema de coloração de vértices em problemas reais de engenharia, incluindo por exemplo problemas de agendamento [Leighton, 1979], programação de horários [de Werra, 1985], plataformas de trens [Caprara et al., 2007], atribuição de frequência [Gamst, 1986] e redes de comunicação [Woo et al., 1991].

A Figura 3.2 ilustra a coloração de vértices de diferentes grafos.

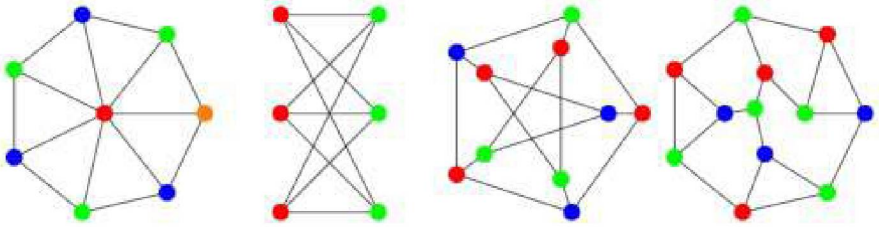


Figura 3.2: Exemplos de coloração de vértices.
Fonte: [Weisstein, 2008]

Definição: O **número cromático** de um grafo G, denotado por $\chi(G)$, é o menor k tal que G admite uma k-coloração. Determinar o número cromático de um grafo, é determinar o menor valor k possível para obter então uma k-coloração do grafo, como exemplificado na Figura 3.3. Este problema de determinação do número cromático de um grafo, também é um problema NP-Difícil [Skiena, 1991].

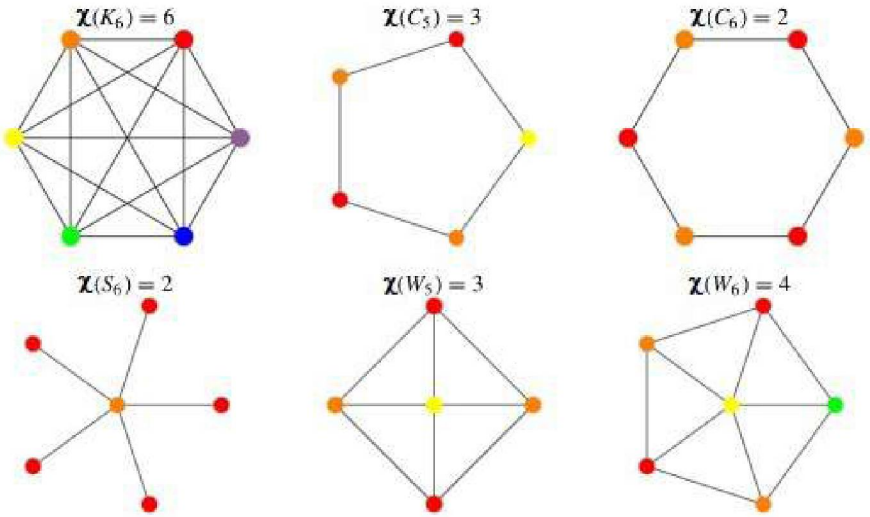


Figura 3.3: Número cromático de diferentes grafos
Adaptado de Weisstein [2008]

Desta maneira, a resposta positiva para o **problema de 3-Coloração**, ou dizer que um grafo é 3-Colorível, significa que seu número cromático é no máximo 3.

A prova de que o problema de 3-coloração de vértices é um problema da classe NP-Completo foi desenvolvida pela primeira vez por Garey et al. [1974].

Além do problema de coloração de mapas, citamos outro exemplo cuja instância pode ser modelada como coloração de grafos [Casselgren, 2011]:

Agendamento de reuniões de comissões

O problema de elaboração da agenda de reuniões de comissões de uma organização pode ser modelado como um problema de coloração de grafos, como segue:

- supondo que uma organização tem um número m de comissões, $C1, ..., Cm$ e precisa organizar reuniões com todas as comissões;
- se uma pessoa participa de várias diferentes comissões, então cada reunião precisa ser agendada em horários diferentes.
- consideramos um grafo G, onde cada vértice representa uma diferente comissão;
- dois vértices do grafo G são conectados por uma aresta se as correspondentes comissões possuírem pelo menos um membro em comum;
- podemos atribuir diferentes cores (representando diferentes horários) para os vértices de G, de maneira que se dois vértices forem unidos por uma aresta, tenham correspondentes cores diferentes.
- desta maneira, as reuniões das comissões podem ser agendadas em k horários, se e somente se, for possível colorir o grafo G com k cores.

No exemplo da figura, ilustramos $m = 8$ comissões (vértices), sendo possível agendar reuniões destas comissões em $k = 4$ horários diferentes (cores), de maneira que os participantes de mais de uma comissão consigam participar de todas as reuniões.

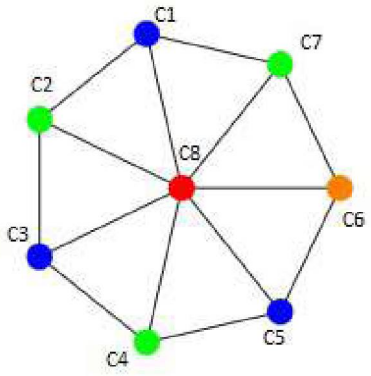


Figura 3.4: Coloração de grafo representando agendamento de reuniões

Além do exemplo do problema de agendamento de reuniões citado na seção anterior, outros diferentes problemas de otimização podem ser modelados como instâncias de problemas de coloração de grafos [Casselgren, 2011].

3.2 MODELAGEM DO PROBLEMA DE 3-COLORAÇÃO

Como dito anteriormente, modelar corretamente um problema de otimização possibilita a determinação da técnica de resolução a ser adotada.

O problema da 3-coloração de grafos pode, então, ser modelado de diferentes maneiras, o que resulta em diferentes abordagens para sua resolução.

Neste trabalho, demonstramos a modelagem do problema de 3-coloração de grafos enquanto instâncias (3,2)-CSP - *Constraint Satisfaction Problem* e enquanto instâncias SAT - Satisfabilidade Booleana.

3.2.1 Constraint-Satisfaction Problem - CSP

Para sua resolução, o problema de 3-coloração de grafos pode ser modelado como uma instância de **problema de satisfação de restrições** (*Constraint-Satisfaction Problem*) – **CSP**, mais especificamente como o problema **(3,2)-CSP**.

De acordo com Lima [2016], uma instância de um problema de satisfação de restrições **(a,b)-CSP** consiste dos seguintes componentes:

- **uma tripla** (X, D, R)**,** onde X, D e R são conjuntos finitos disjuntos. X é denominado conjunto de variáveis, D é o conjunto de valores que serão atribuídos às variáveis, sendo $|D| = a$, e R é o conjunto de restrições. Uma restrição é um par (t, f), onde t é uma b-upla de variáveis e f é uma relação de b valores de D.

- **Resposta:** uma valoração às variáveis de tal forma que todas as restrições sejam satisfeitas. Uma restrição (t, f) é satisfeita quando não acontece de $t_i = f_i$ para todo $i \in \{1, \ldots, b\}$ (todas ao mesmo tempo).

Exemplificando uma instância de um problema de 3-coloração como (3,2)-CSP na Figura 3.5 , onde temos:

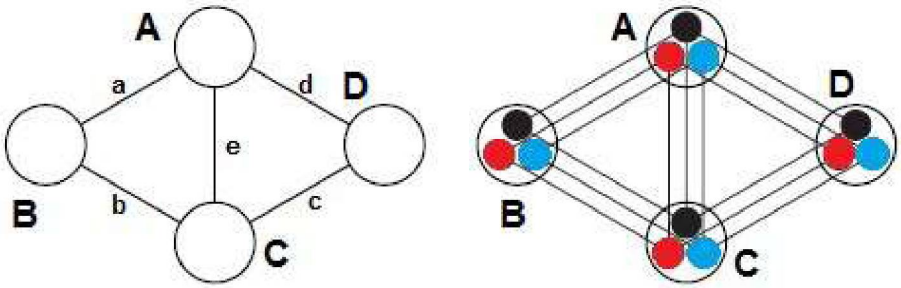


Figura 3.5: Exemplo de 3-coloração como CSP.
Adaptado de Beigel e Eppstein [2005]

- Cada vértice {A, B, C, D} do grafo é uma variável do problema, e está restrita a no máximo 3 valores (cores) do domínio $D = \{0,1,2\}$;
- A cor preta representa o valor 0, a cor vermelha representa o valor 1 e a cor azul representa o valor 2;
- Cada aresta do multigrafo representa uma restrição (cláusula) de incompatibilidade de cores entre cada par de vértices. Assim, para cada aresta do grafo original serão criadas 3 restrições para o problema de 3-coloração:

(A=0, B=0), (A=1, B=1), (A=2, B=2),	*(aresta a)*
(A=0, C=0), (A=1, C=1), (A=2, C=2),	*(aresta e)*
(A=0, D=0), (A=1, D=1), (A=2, D=2),	*(aresta d)*
(B=0, C=0), (B=1, C=1), (B=2, C=2),	*(aresta b)*
(C=0, D=0), (C=1, D=1), (C=2, D=2)	*(aresta c)*

Resposta: Seguem os valores (cores) atribuídos às variáveis (vértices) de maneira a não violar nenhuma das restrições, conforme Figura 3.6:

A = 1,
B = 0,
C = 2,
D = 0

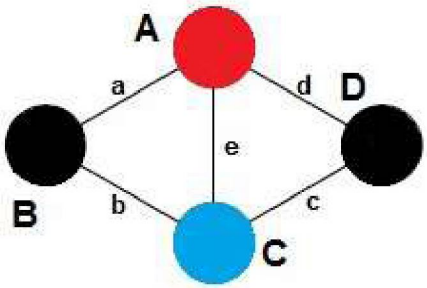


Figura 3.6: Exemplo resolvido de 3-coloração como CSP.
Adaptado de Beigel e Eppstein [2005]

3.2.2 Satisfabilidade Booleana - SAT

Também é possível modelar o problema de 3-coloração como instância de um **problema de Satisfabillidade Booleana (SAT)**.

Definição: um problema SAT visa saber se uma fórmula booleana escrita na forma normal conjuntiva (CNF) pode ser verdadeira. Existindo uma valoração para suas variáveis que torne a fórmula verdadeira, dizemos que ela é satisfatível, caso contrário, ela é insatisfatível.

Segundo Gu [1994], um problema SAT possui três componentes:

- um conjunto de m variáveis: $x1, x2, ..., xm$;
- um conjunto de literais. Um literal é uma variável x, ou uma negação da variável x;
- um conjunto de n cláusulas distintas: $C1, C2, ..., Cn$. Cada cláusula consiste apenas de literais combinados por conectivos lógicos or $(\vee)$.

Com estes componentes, o objetivo de um problema SAT é determinar se existe uma valoração que torne satisfatível a fórmula booleana:

$C1 \wedge C2 \wedge ... \wedge Cn$, onde $\wedge$ é o conectivo lógico and.

Exemplo:

Para exemplificar a modelagem de uma instância do problema de 3-coloração como SAT, ilustramos com um grafo de $n = 4$ **vértices e** $m = 4$ **arestas**, conforme Figura 3.7:

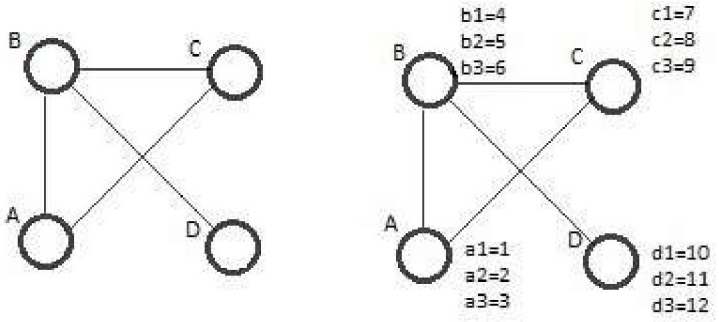


Figura 3.7: Exemplo de 3-coloração como SAT.

- para cada vértice do grafo são criadas outras 3 variáveis, correspondentes aos 3 possíveis valores (cores), conforme tabela 3.2:

Tabela 3.2: Tabela de variáveis 3-coloração como SAT

Vértice	**Variáveis**	**Valor**
A	a1	cor 1
	a2	cor 2
	a3	cor 3
B	b1	cor 1
	b2	cor 2
	b3	cor 3
C	c1	cor 1
	c2	cor 2
	c3	cor 3
D	d1	cor 1
	d2	cor 2
	d3	cor 3

Como condições, ou cláusulas do problema:

- cada vértice do grafo tem **ao menos uma cor**:

  Cláusulas:

  $(a1 \vee a2 \vee a3)$ *(Vértice A = cor 1 ou cor 2 ou cor 3)*
  $(b1 \vee b2 \vee b3)$ *(Vértice B = cor 1 ou cor 2 ou cor 3)*
  $(c1 \vee c2 \vee c3)$ *(Vértice C = cor 1 ou cor 2 ou cor 3)*
  $(d1 \vee d2 \vee d3)$ *(Vértice D = cor 1 ou cor 2 ou cor 3)*

  Portanto esta condição determina **1 cláusula para cada vértice** do grafo.

- cada vértice do grafo tem **no máximo uma cor**:

  Cláusulas:

  $(\sim a1 \vee \sim a2)$ *(Vértice A = não cor 1 ou não cor 2)*
  $(\sim a2 \vee \sim a3)$ *(Vértice A = não cor 2 ou não cor 3)*
  $(\sim a3 \vee \sim a1)$ *(Vértice A = não cor 3 ou não cor 1)*
  $(\sim b1 \vee \sim b2)$ *(Vértice B = não cor 1 ou não cor 2)*
  $(\sim b2 \vee \sim b3)$ *(Vértice B = não cor 2 ou não cor 3)*
  $(\sim b3 \vee \sim b1)$ *(Vértice B = não cor 3 ou não cor 1)*
  $(\sim c1 \vee \sim c2)$ *(Vértice C = não cor 1 ou não cor 2)*
  $(\sim c2 \vee \sim c3)$ *(Vértice C = não cor 2 ou não cor 3)*
  $(\sim c3 \vee \sim c1)$ *(Vértice C = não cor 3 ou não cor 1)*
  $(\sim d1 \vee \sim d2)$ *(Vértice D = não cor 1 ou não cor 2)*
  $(\sim d2 \vee \sim d3)$ *(Vértice D = não cor 2 ou não cor 3)*
  $(\sim d3 \vee \sim d1)$ *(Vértice D = não cor 3 ou não cor 1)*

  Portanto esta condição determina **mais 3 cláusulas para cada vértice** do grafo.

- e, **vértices adjacentes precisam ser de cores diferentes**:

  Cláusulas:

  $(\sim a1 \vee \sim b1)$
  $(\sim a2 \vee \sim b2)$

$(\sim a3 \vee \sim b3)$ *(Par de vértices (A,B))*

$(\sim b1 \vee \sim c1)$
$(\sim b2 \vee \sim c2)$
$(\sim b3 \vee \sim c3)$ *(Par de vértices (B,C))*

$(\sim a1 \vee \sim c1)$
$(\sim a2 \vee \sim c2)$
$(\sim a3 \vee \sim c3)$ *(Par de vértices (A,C))*

$(\sim b1 \vee \sim d1)$
$(\sim b2 \vee \sim d2)$
$(\sim b3 \vee \sim d3)$ *(Par de vértices (B,D))*

Portanto, são **3 cláusulas diferentes para cada aresta** do grafo.

Desta maneira, conforme este exemplo, o problema de 3-coloração com n vértices e m arestas terá sua representação enquanto SAT como um problema com $3n$ variáveis (demonstrado na tabela 3.2) e, somando todas as condições definidas acima, $4n + 3m$ cláusulas.

Em nosso exemplo, o grafo de $n = 4$ vértices e $m = 4$ arestas, temos **12 variáveis e 28 cláusulas**.

Resposta: Conforme Figura 3.8, uma valoração que torna safisfatível a fórmula booleana composta pela conjunção de todas as cláusulas é:

variável $a1 = 1$, *(Vértice A = cor 1)*
variável $b2 = 1$, *(Vértice B = cor 2)*
variável $c3 = 1$, *(Vértice C = cor 3)*
variável $d1 = 1$ *(Vértice D = cor 1)*

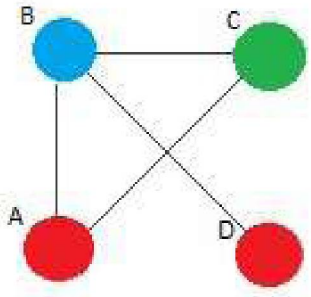


Figura 3.8: Exemplo resolvido de 3-coloração como SAT.

4 RESOLUÇÃO DO PROBLEMA DE 3-COLORAÇÃO COM O SCIP

4.1 SCIP

SCIP Optimization Suite consiste em um conjunto de softwares projetados para modelar e resolver diversos tipos de problemas de otimização.

Atualmente é um dos mais rápidos resolvedores de problemas de programação linear inteira mista (MILP) e programação não-linear inteira mista (MINLP) [Berlin, 2021b].

Algumas das partes que compõem o SCIP Optimization Suite são as seguintes [Berlin, 2021b]:

- ZIMPL (Zuse Institute Mathematical Programming Language) - linguagem de modelagem usada para traduzir um modelo matemático de problema em um programa linear ou não linear. O programa linear ou não linear será escrito no formato .lp ou .mps, para então ser resolvido [Berthold et al., 2012];
- SoPlex (Sequencial object-oriented simPlex) - resolvedor padrão de problemas de programação linear dentro do SCIP. É uma implementação avançada do Método Simplex Revisado. Detalhes sobre o Simplex Revisado em [Papadimitriou e Steiglitz, 1998] e também em [Matousek e Gärtner, 2007]. Realiza etapas de pré-processamento (presolving) antes de aplicar o algoritmo simplex [Berlin, 2021c].
- GCG (Generic Column Generation) - é um algoritmo de branch-cut-and-price [Gamrath et al., 2020], com uma implementação automática da técnica de decomposição de Dantzig-Wolfe [Dantzig e Wolfe, 1960].

4.2 TIPOS DE PROBLEMAS RESOLVIDOS PELO SCIP

O SCIP como resolvedor de problemas de programação linear inteira mista (MILP) e programação não-linear inteira mista (MINLP), traz em sua documentação uma lista (não exaustiva) de tipos de problemas de otimização e recomendações para sua compilação.

Na tabela abaixo listamos alguns destes tipos de problemas e respectivos formatos de arquivos suportados para entrada dos dados do problema a ser resolvido pelo SCIP:

Tipo de Problema	**Formato de Arquivo Suportado**
Programação Linear Inteira Mista	CIP / MPS / LP / ZPL
Programação Não-Linear Inteira Mista	CIP / GMS / OSiL / PIP /ZPL
Programação Linear	MIP
Otimização Pseudobooleana	WBO / OPB
Satisfabilidade Booleana (SAT)	CNF

A maneira de carregar um problema de otimização para resolução com o SCIP é através de arquivos em formatos que a ferramenta consegue analisar diretamente.

Demonstraremos especificamente como modelar o problema de 3-coloração no formato do arquivo suportado enquanto SAT, bem como um exemplo de sua resolução com o SCIP, na subseção a seguir.

4.2.1 Formato de arquivo - CNF

Para a resolução do problema de 3-coloração, o SCIP utiliza arquivos no formato CNF para receber o grafo de entrada. Este formato define as restrições do problema como expressões booleanas, escritas na forma normal conjuntiva [Burkardt, 2008].

Usando como exemplo o grafo da Figura 3.7, que possui $n = 4$ vértices e $m = 4$ arestas, segue abaixo o conteúdo de seu respectivo arquivo CNF, a ser processado pelo SCIP:

Linhas	Conteúdo	Linhas	Conteúdo
1	c 4 vertices	25	-4 -5 0
2	c 4 edges	26	-5 -6 0
3	c 12 variables	27	-6 -4 0
4	c 28 clauses	28	7 8 9 0
5	c variables meaning:	29	-7 -8 0
6	c var 1 –> c(1) = 0	30	-8 -9 0
7	c var 2 –> c(1) = 1	31	-9 -7 0
8	c var 3 –> c(1) = 2	32	10 11 12 0
9	c var 4 –> c(2) = 0	33	-10 -11 0
10	c var 5 –> c(2) = 1	34	-11 -12 0
11	c var 6 –> c(2) = 2	35	-12 -10 0
12	c var 7 –> c(3) = 0	36	-1 -4 0
13	c var 8 –> c(3) = 1	37	-2 -5 0
14	c var 9 –> c(3) = 2	38	-3 -6 0
15	c var 10 –> c(4) = 0	39	-1 -7 0
16	c var 11 –> c(4) = 1	40	-2 -8 0
17	c var 12 –> c(4) = 2	41	-3 -9 0
18	c	42	-4 -7 0
19	p cnf 12 28	43	-5 -8 0
20	1 2 3 0	44	-6 -9 0
21	-1 -2 0	45	-4 -10 0
22	-2 -3 0	46	-5 -11 0
23	-3 -1 0	47	-6 -12 0
24	4 5 6 0		

Neste arquivo, as linhas **1 a 18, iniciadas por "c"**, correspondem a comentários usados para caracterizar o problema.

Em seguida, a linha **19, iniciada por "p"** traz as informações do problema, indicando que é um arquivo do tipo CNF, seguido do número de variáveis e de cláusulas do problema.

As demais linhas, **20 a 47, são as cláusulas do problema**, uma a uma. Cada cláusula é formada pelo índice da correspondente variável, ou negação desta variável. A negação das variáveis é indicada pelo sinal negativo. Cada cláusula é uma combinação de variáveis e termina com 0.

Na interpretação destas cláusulas, os índices são combinados por conectivos lógicos *or* e as cláusulas combinadas pelo conectivo lógico *and*, formando a fórmula booleana para a qual se busca a valoração que a torne verdadeira.

Na Figura 4.1, exemplificamos o processamento do arquivo CNF correspondente à Figura 3.7.

```
============

original problem has 12 variables (12 bin, 0 int, 0 impl, 0 cont) and 28 constraints

presolving:
  (0.0s) running MILP presolver
  (0.0s) MILP presolver found nothing
  (0.0s) probing cycle finished: starting next cycle
  (0.0s) symmetry computation started: requiring (bin +, int -, cont +), (fixed: bin -, int +, cont -)
  (0.0s) symmetry computation finished: 3 generators found (max: 1500, log10 of symmetry group size: 1.1)
presolving (1 rounds: 1 fast, 1 medium, 1 exhaustive):
0 deleted vars, 0 deleted constraints, 0 added constraints, 0 tightened bounds, 0 added holes, 0 changed sides, 0 changed
coefficients
0 implications, 24 cliques
presolved problem has 12 variables (12 bin, 0 int, 0 impl, 0 cont) and 28 constraints
   28 constraints of type <logicor>
transformed objective value is always integral (scale: 1)
Presolving Time: 0.01

time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|strbr|  dualbound   | primalbound
|  gap   | compl.
p 0.0s|    1 |    0 |    0 |    - | clique|  0 |  12 |  28 |  28 |  0 |  0 |  0 |  0 | 0.000000e+00 | 0.000000e+00 |   0.00%|
unknown

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.02
Solving Nodes      : 1
Primal Bound       : +0.00000000000000e+00 (1 solutions)
Dual Bound         : +0.00000000000000e+00
Gap                : 0.00 %


objective value:                 0
x1                               1   (obj:0)
x5                               1   (obj:0)
x9                               1   (obj:0)
x10                              1   (obj:0)
```

Figura 4.1: Exemplo de resolução com o SCIP.

Neste exemplo, a solução é:

$x_1 = 1$;

$x_5 = 1$;

$x_9 = 1$;

$x_{10} = 1$

Esta solução corresponde aos vértices e cores:

1 - cor 0;

2 - cor 1;

3 - cor 2;

4 - cor 0

Exemplos de resolução de diversos problemas com o SCIP podem ser consultados em sua documentação [Berlin, 2021b], bem como exemplos de projetos em que se utiliza o SCIP, desenvolvidos por outros autores [Berlin, 2021a, Related Works].

5 EXPERIMENTO

Como mencionado anteriormente na revisão de algoritmos para resolução do problema de coloração de grafos [Lima, 2016], algumas rotinas de pré-processamento podem reduzir o tempo de máquina necessário para se chegar à resposta do problema abordado.

No caso específico do problema de 3-coloração, podemos considerar as características e propriedades dos grafos para propor uma redução nas instâncias a serem efetivamente processadas pelo resolvedor, com o objetivo de reduzir o tempo de resolução do problema.

A redução proposta é simples, e apenas uma das possíveis abordagens que podem ser trabalhadas com o objetivo de reduzir a quantidade de restrições a serem processadas.

5.1 ROTINA DE PRÉ-PROCESSAMENTO

A rotina de pré-processamento para resolução do problema de 3-coloração que propomos é realizada nas instâncias enquanto grafo.

Considerando:

- **vértices de grau menor que 3** - se um grafo G possui vértice v de grau menor que 3, podemos primeiramente colorir o restante deste grafo G, e depois atribuir a v uma das cores restantes, diferente das cores atribuídas aos vizinhos de v.

  Na Figura 5.1, temos um grafo G, em que os vértices 2 e 3 são de grau 2 (a). Desta maneira, colorimos primeiramente os demais vértices do grafo (b), e depois atribuímos aos vértices 2 e 3, as cores restantes e diferentes da cor de seus vértices vizinhos (c).

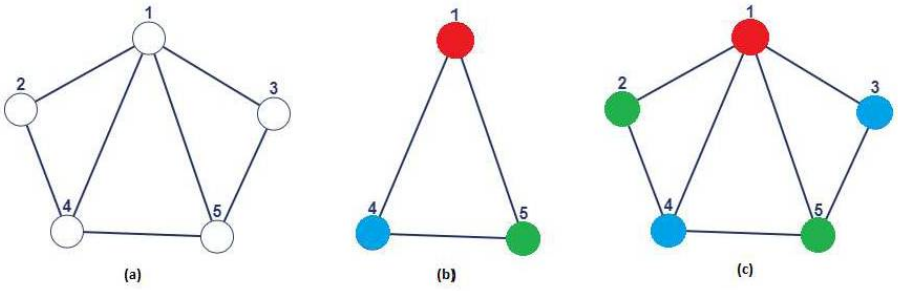


Figura 5.1: Grafo com vértices de grau menor que 3.

- **grafo diamante** é um grafo simples que contém 4 vértices e 5 arestas. É um grafo completo tripartido ($K_{1,1,2}$), ou ainda um K_4 sem uma das arestas ($K_4 - e$), de número cromático 3, conforme Figura 5.2.

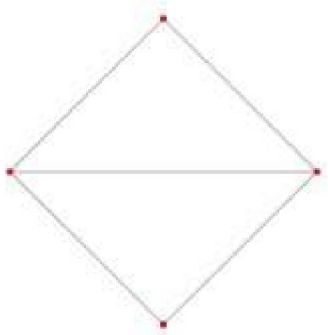


Figura 5.2: Grafo diamante.
Fonte: Weisstein [2008]

Em uma 3-coloração, os vértices de grau 2 de um grafo diamante serão coloridos com a mesma cor, como ilustrado em 5.3.

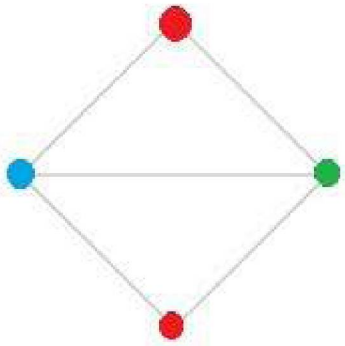


Figura 5.3: Grafo diamante colorido.
Adaptado de Weisstein [2008]

- **na modelagem de um problema de 3-coloração de um grafo como SAT**, o número de variáveis e cláusulas, ou condições do problema a serem satisfeitas, está relacionado com o número de vértices e arestas do grafo. Para um grafo G de n vértices e m arestas, serão atribuídas $3n$ variáveis e $4n + 3m$ cláusulas para serem processadas na resolução do problema, conforme detalhado no tópico 3.2.

Ao remover um dos vértices não adjacentes dos subgrafos diamantes e demais vértices de grau 0, 1 ou 2 do grafo original, reduzimos o número de variáveis e cláusulas a serem efetivamente processadas pelo SCIP, para resolução do problema.

Exemplo:

Abaixo ilustramos uma instância com 10 vértices e 38 arestas em que a rotina de pré-processamento proposta remove todos os vértices.

A instância original, antes da rotina de pré-processamento está representada pela Figura 5.4.

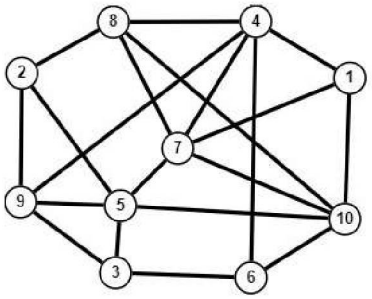


Figura 5.4: Instância original

Na Figura 5.5 (A), a rotina de pré-processamento identifica o diamante formado pelos vértices 1, 4, 7 e 10 e seus respectivos vértices não adjacentes 4 e 10. Para estes vértices não adjacentes, a cor a ser atribuída será necessariamente a mesma, então estes podem ser unificados, mantendo os vértices vizinhos a serem coloridos.

Na sequência, em (B), ilustramos a instância após a unificação dos vértices do diamante identificado.

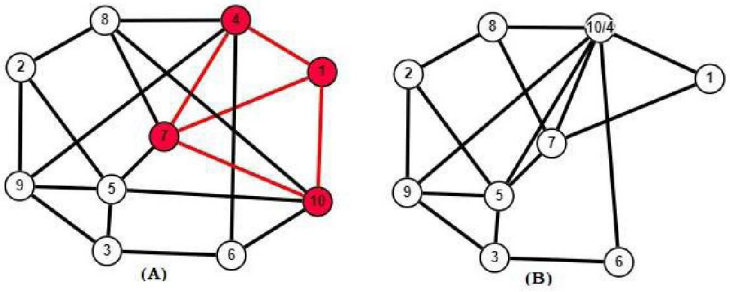


Figura 5.5: Identificação do diamante 1,4,7,10 na execução da rotina de pré-processamento.

Em seguida a rotina identifica o diamante 2, 3, 5 e 9 - Figura 5.6 (A); e unifica os vértices 2 e 3, não adjacentes do diamante - Figura 5.6 (B).

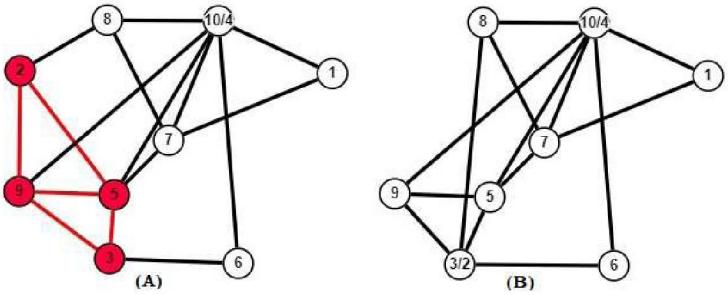


Figura 5.6: Identificação do diamante 2,3,5,9 na execução da rotina de pré-processamento.

Desta maneira, a rotina segue identificando todos os demais diamantes desta instância, conforme as figuras 5.7 a 5.10.

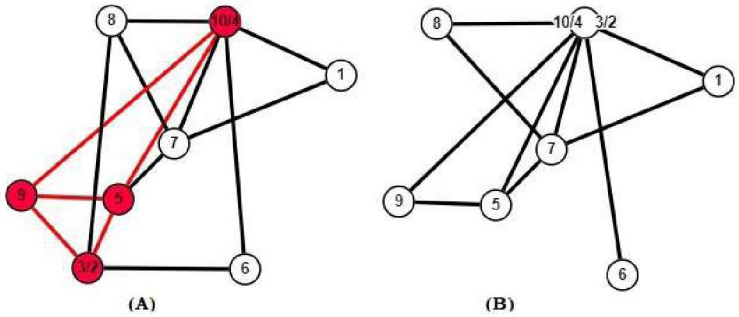


Figura 5.7: Identificação do diamante 3/2,5,9,10/4 na execução da rotina de pré-processamento.

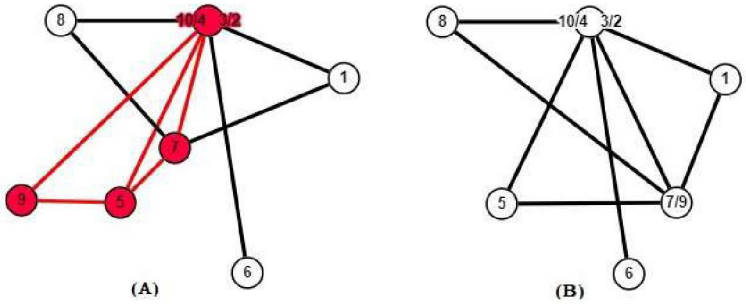


Figura 5.8: Identificação do diamante 5,7,9,10/4/3/2 na execução da rotina de pré-processamento.

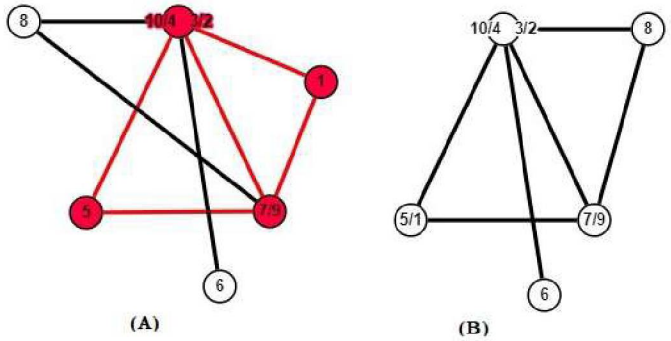


Figura 5.9: Identificação do diamante 1,5,7/9,10/4/3/2 na execução da rotina de pré-processamento.

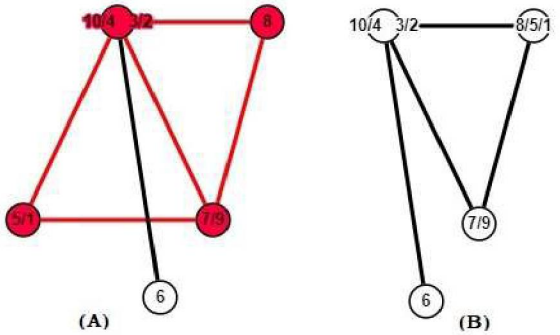


Figura 5.10: Identificação do diamante 5/1,8,7/9,10/4/3/2 na execução da rotina de pré-processamento.

Após a identificação de todos os diamantes da instância, a rotina faz a identificação e remoção dos vértices de grau menor do que 3, conforme início ilustrado na Figura 5.11.

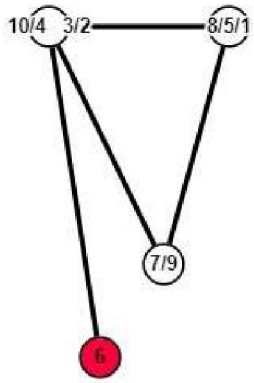


Figura 5.11: Identificação de vértice de grau menor que 3 na execução da rotina de pré-processamento.

Estes vértices de grau menor que 3 são removidos e rotulados, e posteriormente, após colorir o restante do grafo, a estes será atribuída uma das cores restantes, diferente das cores atribuídas aos seus vizinhos.

Após a identificação, remoção e rotulação dos vértices que atendem às especificidades da rotina, havendo algum subgrafo restante, a instância correspondente será entregue em formato CNF para resolução pelo SCIP.

Com esta instância colorida pelo SCIP, faz-se a coloração dos vértices anteriormente removidos pela rotina, considerando as cores já atribuídas aos seus vizinhos.

Os algoritmos abaixo correspondem à execução da rotina de pré-processamento chamada Descasca - Algoritmo 1, identificação de diamantes - Algoritmo 2, identificação de P_3 - Algoritmo 3 (algoritmo auxiliar na identificação de diamantes):

Algoritmo 1: Descasca

```
Entrada: Um grafo G = (V, E)
Saída  : Um grafo G'
G' ← G
removeu ← True
Enquanto removeu = True
    removeu ← False
    /* Remove vértices de grau menor ou igual a 2:      */
    removeu-graumenor = True
    Enquanto removeu-graumenor = True
        removeu-graumenor ← False
        Para u ∈ V
            Se d_G'(u) ≤ 2
                remove vértice u de G'
                removeu ← True
                removeu-graumenor ← True
    /* Identifica os diamantes:                         */
    Enquanto Encontra-diamante(G', w, x, y, z)
        unifica os vértices w e z em G'
        removeu ← True
```

Algoritmo 2: Encontra-diamante(G, w, x, y, z)

```
Entrada: Um grafo G
Saída  : True, caso exista um diamante em G. Neste caso w, x, y e z são vértices
         tais que (w, x, y, z) forma um diamante com w e z não adjacentes. False,
         caso não exista.
Para v ∈ V(G)
    /* G[X] é o grafo induzido pelo conjunto X          */
    /* N(v) é o conjunto dos vizinhos de v              */
    Se Encontra-p3(G[N(v)], a, b, c)
        (w, x, y, z) ← (a, v, b, c)
        Devolva True
Devolva False
```

Algoritmo 3: Encontra-p3(H, a, b, c)

```
Entrada: Um grafo H
Saída  : True, caso exista um P_3 em H. Neste caso a, b e c são vértices tais que
         (a, b, c) forma um P_3. False, caso não exista.
Faça uma busca em largura em H e calcule as componentes conexas C_i
Para cada componente C_i
    r ← raíz da árvore da busca em largura da componente C_i
    Se a busca encontrou um vértice w a profundidade 3
        x ← pai de w
        (a, b, c) ← (r, x, w)
        Devolva True
    Senão
        Para w ∈ C_i, com w ≠ r
            /* d_H(w) é o grau de w em H                                  */
            Se d_H(w) < |C_i| − 1
                Encontre um vértice x em C_i tal que {w, x} não é aresta de H
                (a, b, c) ← (w, r, x)
                Devolva True
Devolva False
```

Disponibilizamos o código completo em diretório público[1].

Desta maneira, executando esta rotina de pré-processamento nas instâncias selecionadas e detalhadas nas próximas seções, neste experimento buscamos responder quão efetivo seria realizar um pré-processamento na instância enquanto grafo para a redução do tempo necessário para a resolução do problema pelo SCIP.

5.2 INSTÂNCIAS UTILIZADAS

Para o desenvolvimento deste trabalho utilizamos instâncias 3-coloríveis com diferentes quantidades de vértices e arestas. Entre as diferentes instâncias, analisamos exemplos em que a rotina de pré-processamento proposta efetivamente reduz as variáveis e cláusulas a serem processadas pelo SCIP. Chamaremos estas de instâncias "Descascáveis".

Também utilizaremos instâncias em que a rotina de pré-processamento não é efetiva, ou seja, instâncias que não possuem diamantes e/ou vértices de grau menor que 3, chamadas de instâncias "Não-Descascáveis".

5.2.1 Instâncias Descascáveis

As instâncias "Descascáveis"utilizadas, ou seja, instâncias que possuem diamantes e vértices de grau menor do que 3, são instâncias 3-coloríveis geradas randomicamente através de um gerador de instâncias k-partidas. No caso, geramos instâncias 3-partidas de grafos com n vértices, em formato Graphviz - extensão gv [Graphviz, 2022].

[1]Disponível em: hhttps://gitlab.c3sl.ufpr.br/teoria/vfbelli-3color

O código do gerador de instâncias mencionado está disponível em diretório público[2].

Algoritmo 4: Gerador de Instâncias

Entrada: Dois números inteiros: k (número de partes), n (número de vértices)
Saída : Um grafo k-partido G, com n vértices
$p[]$ /* tamanho de cada parte */
$parte[]$ /* parte de cada vértice */
$produto \leftarrow 0$
/* aleatoriamente distribui vértices nas k-partes */
Enquanto $produto = 0$
    distribui vértices randomicamente nas partes criadas calculando o tamanho $p[i]$ de cada parte i
    $produto = \prod_{i=1}^{k} p[i]$
    /* se $produto = 0$, alguma parte ficou vazia */
gera um grafo G sem arestas
/* cria arestas entre as partes */
Para u *de* 1 *a* $n-1$
    Para v *de* $u+1$ *a* n
        Se $parte[u] \neq parte[v]$
            com probabilidade $\frac{1}{2}$ cria aresta entre u e v em G

Devolva G *em arquivo no formato* gv

Para o experimento, utilizamos instâncias com $n = \{10, 20, 40, 80, 160, 320, 640\}$ vértices.

Nestas instâncias, a rotina de pré-processamento do experimento remove todos os vértices do grafo original e o arquivo CNF a ser processado pelo SCIP não possui variáveis ou cláusulas. Desta maneira, a resolução da 3-coloração destas instâncias é feita através da correspondente identificação dos vértices não adjacentes dos diamantes e vértices de grau menor que 3, realizada anteriormente.

A Figura 5.12 ilustra uma instância 3-colorível, com 10 vértices, gerada randomicamente. Esta instância do exemplo possui diamantes a serem identificados pela rotina de pré-processamento. A identificação dos diamantes, unificação de seus vértices não adjacentes e posterior remoção de vértices de grau menor que 3 está demonstrada na seção 5.1.

[2]Disponível em: https://gitlab.c3sl.ufpr.br/teoria/vfbelli-3color

```
strict graph "random-3-partite-4-2-4" {
1;
2;
3;
4;
5;
6;
7;
8;
9;
10;
1 -- 4;
1 -- 7;
1 -- 10;
2 -- 5;
2 -- 8;
2 -- 9;
3 -- 5;
3 -- 6;
3 -- 9;
4 -- 6;
4 -- 7;
4 -- 8;
4 -- 9;
5 -- 7;
5 -- 9;
5 -- 10;
6 -- 10;
7 -- 8;
7 -- 10;
8 -- 10;
}
```

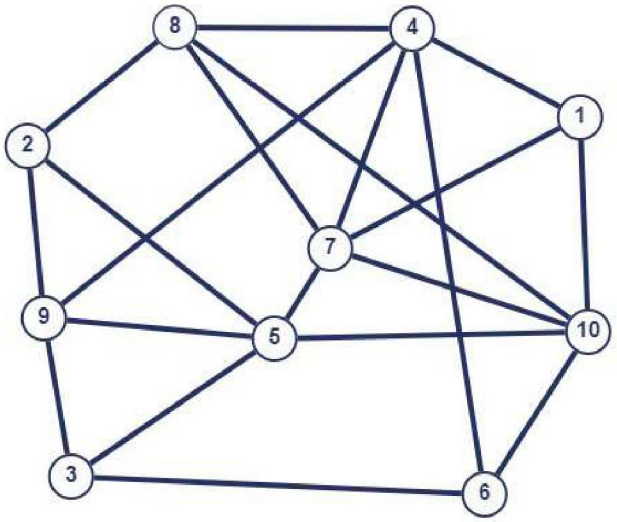


Figura 5.12: Instância Random 3-colorível, 10 vértices

5.2.2 Instâncias Não-Descascáveis

Como instâncias "Não-Descascáveis", utilizamos as instâncias MUG_{nt}, também chamadas 4-critical, propostas por Mizuno e Nishihara [2008].

Definição: Um grafo G, será 4-critical se possuir número cromático 4, e qualquer subgrafo G' ($G' \subset G$) for 3-colorível.

O subgrafo G', 3-colorível, é obtido através da remoção arbitrária de alguma aresta de vértice de grau maior que 3, de G.

Exemplo do menor grafo 4-critical possível é o K_4 (Figura 5.13 (a)), pois é um grafo não-3-colorível tal que ao se remover uma das arestas, o subgrafo resultante é 3-colorível (Figura 5.13 (b), na figura chamado de *n4c*).

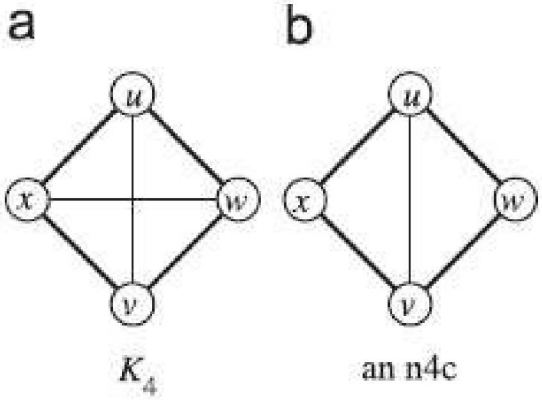


Figura 5.13: Menor grafo 4-critical
Fonte: Beigel e Eppstein [2005]

Utilizando esta definição de 4-critical, através de tentativa e erro, os autores geraram 7 diferentes instâncias MUG_{nt} (MUG: *"minimal unsolvable graphs"*), onde n significa o número de vértices e t foi utilizado para identificar o tipo da instância, se necessário. Na Figura 5.14, detalhamos as instâncias MUG_{nt} criadas, onde n é número de vértices e m é o número de arestas destas.

MUG_{nt}	n	m
MUG9	9	16
MUG10	10	18
MUG11a	11	20
MUG11b	11	19
MUG12a	12	22
MUG12b	12	22
MUG12c	12	21

Figura 5.14: Instâncias MUG_{nt}
Fonte: Beigel e Eppstein [2005]

Na Figura 5.15 ilustramos as instâncias $MUG9$ e $MUG10$.

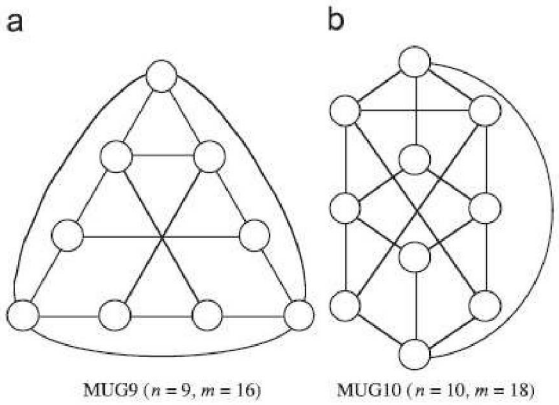


Figura 5.15: Instâncias $MUG9$ e $MUG10$
Fonte: Beigel e Eppstein [2005]

Uma particularidade das instâncias MUG_{nt} propostas, é que estas possuem apenas vértices de grau 3, 4 ou 5. A instância utilizada para o experimento foi a MUG_10, que além de não possuir vértices de grau menor que 3, também não possui diamantes, portanto a rotina de pré-processamento que propomos não remove nenhum vértice desta instância.

Para utilização destas instâncias na rotina de pré-processamento, removemos arbitrariamente uma de suas arestas, de um vértice de grau maior que 3, tornando assim a instância 3-colorível.

Na figura 5.16, ilustramos a instância MUG_{10}, indicando a aresta a ser removida, em seu arquivo *gv* e representação gráfica. E, na figura 5.17, a instância 3-colorível, com a aresta removida.

```
strict graph "MUG10" {
1;
2;
3;
4;
5;
6;
7;
8;
9;
10;
1 -- 2;
1 -- 3;
1 -- 10;
2 -- 3;
2 -- 5;
2 -- 9;
3 -- 6;
3 -- 8;
4 -- 1;
4 -- 5;
4 -- 6;
5 -- 8;
5 -- 7;
6 -- 7;
6 -- 9;
7 -- 10;
8 -- 10;
9 -- 10;
}
```

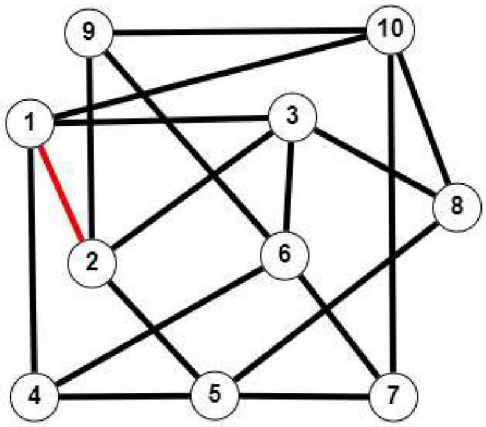


Figura 5.16: Instância MUG_{10}

```
strict graph "MUG10SA" {
1;
2;
3;
4;
5;
6;
7;
8;
9;
10;
1 -- 3;
1 -- 10;
2 -- 3;
2 -- 5;
2 -- 9;
3 -- 6;
3 -- 8;
4 -- 1;
4 -- 5;
4 -- 6;
5 -- 8;
5 -- 7;
6 -- 7;
6 -- 9;
7 -- 10;
8 -- 10;
9 -- 10;
}
```

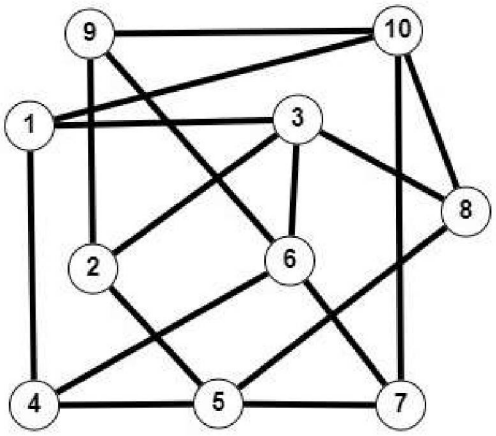


Figura 5.17: Instância MUG_{10} com uma aresta removida

Com o objetivo de obtermos instâncias maiores, replicamos a instância MUG_{10}, com a aresta removida, criando assim instâncias 3-coloríveis com n vértices, em formato Graphviz. Para replicar estas instâncias, simplesmente acrescentamos ao arquivo gv da instância original uma cópia desta, com uma nova numeração dos vértices e arestas, de maneira que para o experimento tenhamos instâncias com $n = \{10, 20, 40, 80, 160, 320, 640\}$ vértices. Na figura 5.18, ilustramos uma destas instâncias, com $n = 20$ vértices.

```
strict graph "MUG10SA_20" {
1;
2;
3;
4;
5;
6;
7;
8;
9;
10;
11;
12;
13;
14;
15;
16;
17;
18;
19;
20;
1 -- 3;
1 -- 10;
2 -- 3;
2 -- 5;
2 -- 9;
3 -- 6;
3 -- 8;
4 -- 1;
4 -- 5;
4 -- 6;
5 -- 8;
5 -- 7;
6 -- 7;
6 -- 9;
7 -- 10;
8 -- 10;
9 -- 10;
11 -- 13;
11 -- 20;
12 -- 13;
12 -- 15;
12 -- 19;
13 -- 16;
13 -- 18;
14 -- 11;
14 -- 15;
14 -- 16;
15 -- 18;
15 -- 17;
16 -- 17;
16 -- 19;
17 -- 20;
18 -- 20;
19 -- 20;
}
```

Figura 5.18: Instância MUG_{10} duplicada

5.2.3 Instâncias Parcialmente Descascáveis

Através da unificação de instâncias "Descascáveis" e instâncias "Não-Descascáveis" criadas anteriormente, pudemos criar novas instâncias em que a rotina de pré-processamento

proposta reduz parcialmente o número de vértices do grafo original, de maneira a entregar para o SCIP uma instância com menor número de variáveis e cláusulas a serem processadas.

As novas instâncias, "Parcialmente Descascáveis" utilizadas neste experimento estão detalhadas na Figura 5.19 abaixo:

	Instância (Grafo) Original			Rotina de pré-processamento		Instância (Grafo) entregue para o SCIP			
	Instância	Número de vértices	Número de aretas	Vértices Removidos	Arestas Removidas	Número de vértices	Número de arestas	Variáveis CNF	Cláusulas CNF
(1)	random3_10_1_MUG10SA_1.gv	20	38	10	21	10	17	30	91
(2)	random3_10_1_MUG10SA_2.gv	30	56	10	21	20	35	60	185
(3)	random3_20_1_MUG10SA_2.gv	40	105	20	71	20	34	60	182
(4)	random3_40_1_MUG10SA_40.gv	80	332	40	264	40	68	120	364
(5)	random3_80_1_MUG10SA_80.gv	160	1141	80	1005	80	136	240	728
(6)	random3_160_1_MUG10SA_160.gv	320	4504	160	4232	160	272	480	1456
(7)	random3_320_1_MUG10SA_320.gv	640	17761	320	17186	320	575	930	3005
(8)	random3_640_1_MUG10SA_640.gv	1280	69508	640	68420	640	1088	1920	5824

Figura 5.19: Instâncias "Parcialmente Descascáveis"

5.3 RESULTADOS

Utilizando as instâncias "Parcialmente Descascáveis" criadas anteriormente, verificamos a eficácia da rotina de pré-processamento proposta neste experimento.

Primeiramente, após criadas as instâncias enquanto grafos, geramos os correspondentes arquivos em formato CNF e executamos o SCIP sem a realização da rotina de pré-processamento, com o objetivo de registrar o tempo de processamento necessário para o SCIP resolver a 3-coloração destas instâncias.

Na Figura 5.20, tabelamos os tempos de processamento do SCIP para cada instância utilizada no experimento.

Instância (Grafo) Original				Instância (Grafo) entregue para o SCIP após rotina de pré-processamento		
Instância	Número de vértices	Número de arestas	Tempo SCIP (segundos)	Número de vértices	Número de arestas	Tempo SCIP (segundos)
(1)	20	38	0,15	10	17	0,04
(2)	30	56	0,04	20	35	0,03
(3)	40	105	0,05	20	34	0,03
(4)	80	332	0,07	40	68	0,04
(5)	160	1141	0,26	80	136	0,14
(6)	320	4504	0,58	160	272	0,19
(7)	640	17761	12,61	320	575	0,10
(8)	1280	69508	6,57	640	1088	0,91

Figura 5.20: Tempos de execução - Instâncias "Parcialmente Descascáveis"

Em seguida, em cada uma das instâncias, executamos a rotina de pré-processamento. Desta maneira, as instâncias a serem entregues para resolução pelo SCIP, foram reduzidas de acordo com o número de vértices removidos pela rotina. Para estas novas instâncias, grafos com

número menor de vértices e arestas, geramos os correspondentes arquivos CNF e executamos novamente o SCIP. Os tempos necessários para processamento das novas instâncias foram registrados e estão demonstrados na tabela da Figura 5.20.

5.3.1 Discussão dos Resultados

De acordo com a tabela da Figura 5.20, na execução de todas as instâncias houve redução do tempo de processamento pelo SCIP, antes e após a rotina de pré-processamento, pois as instâncias entregues após a rotina são grafos com menor número de vértices e arestas. Consequentemente, o número de variáveis e cláusulas dos arquivos CNF a serem processados também é menor.

No gráfico da Figura 5.21, ilustramos a diferença de tempo necessário para o SCIP processar algumas das instâncias, antes e após a rotina de pré-processamento deste experimento.

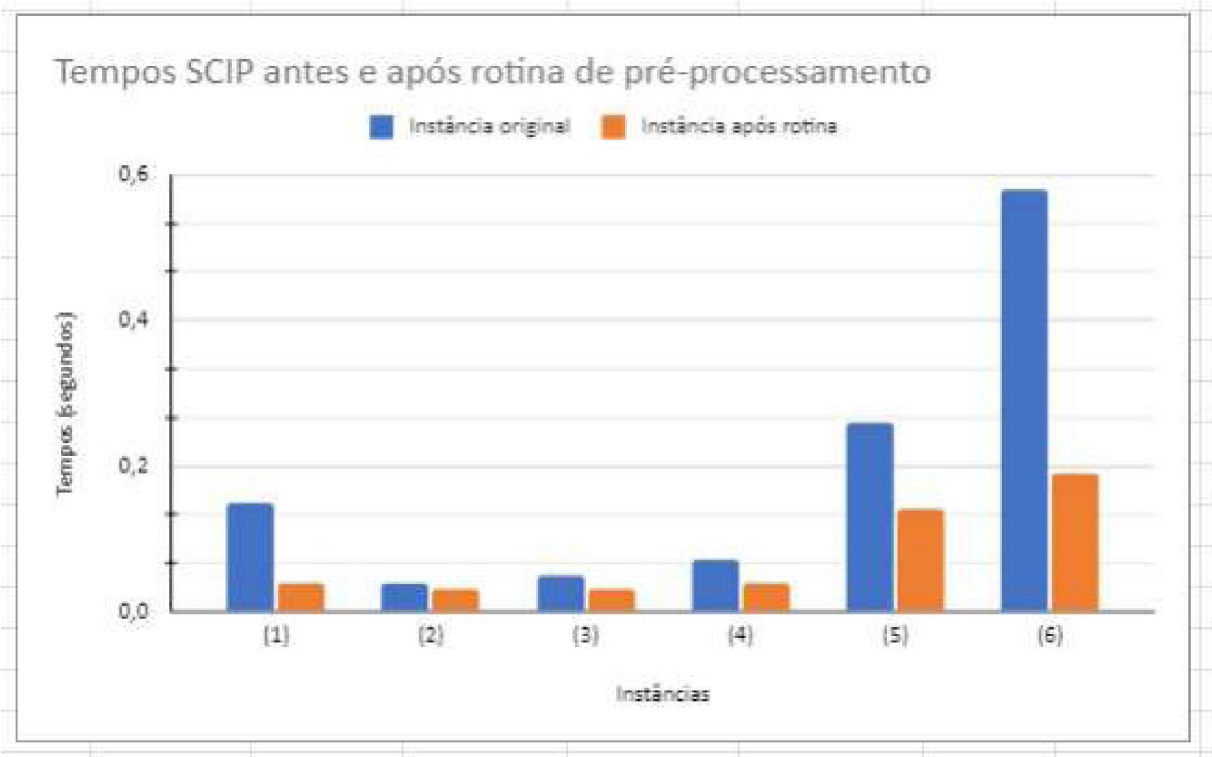


Figura 5.21: Tempos SCIP - antes e após pré-processamento

Importante observar que nestes resultados não consideramos o tempo necessário para gerar os arquivos CNF das instâncias antes e após a execução da rotina de pré-processamento, pois este é um procedimento que se faz necessário independe do experimento, para que o SCIP consiga resolver o problema.

Também não consideramos o tempo necessário para a execução da rotina de pré-processamento completa, desde a identificação, remoção e rotulação dos vértices de diamantes ou de grau menor que 3, até a sua finalização com a correspondente coloração destes vértices identificados e removidos. Apenas consideramos os tempos de execução do SCIP e esta limitação será justificada posteriormente.

Desta maneira, para este tipo de instâncias em que é possível reduzir o número de vértices com a rotina de pré-processamento, esta mostra-se eficiente. Pois, desta maneira conseguimos reduzir o tempo de processamento necessário para o SCIP resolver a 3-coloração destes grafos.

No entanto, ainda foram identificadas outras limitações no experimento, as quais discutiremos em seguida.

5.3.2 Limitações

A rotina de pré-processamento deste experimento é uma proposta bastante simples e apenas uma das possibilidades para redução das instâncias enquanto grafo. Esta rotina mostra-se eficiente quanto à redução das instâncias, pois a existência de diamantes e vértices de grau menor que 3 é comum em todas as instâncias 3-coloríveis geradas para o experimento; com exceção das instâncias MUG_{nt}, escolhidas por não serem reduzidas pela rotina.

Porém, apesar de sua simplicidade, da maneira como foi desenvolvida, a rotina de pré-processamento não mostrou-se eficiente para a redução total do tempo necessário para a resolução do problema de 3-coloração, considerando o tempo necessário para o pré-processamento antes de entregar a instância para o SCIP.

O SCIP é um resolvedor bastante eficiente, capaz de processar instâncias do problema de 3-coloração até maiores que as trabalhadas neste experimento. Nossa rotina de pré-processamento foi desenvolvida em Python, que não é a linguagem mais eficiente, considerando o objetivo do experimento. Também, trabalhamos com a biblioteca Networkx e com as instâncias em formato Graphviz, o que tornou a aplicação mais lenta.

Não trabalhamos com instâncias relativamente grandes, de maneira a verificar o limite de processamento do SCIP, pois o tempo necessário para executar a rotina de pré-processamento em instâncias maiores do que as demonstradas no experimento é inviável.

Para uma rotina de pré-processamento eficaz em sua totalidade, será necessário repensar o seu desenvolvimento, considerando o tempo de execução necessário para a identificação dos diamantes e vértices de grau menor que 3.

6 CONSIDERAÇÕES FINAIS

Com a execução deste experimento, concluímos que a rotina de pré-processamento proposta mostra-se eficiente enquanto redução das instâncias a serem processadas pelo SCIP, de maneira que o tempo necessário para a resolução dos problemas também será reduzido.

Porém, esta rotina precisa ser aprimorada para que seja efetivamente eficaz na redução total do tempo necessário para a resolução do problema de 3-coloração utilizando o SCIP.

Com o objetivo de tornar a mesma proposta de pré-processamento mais eficiente, pode-se repensar a maneira como os diamantes das instâncias são identificados, buscando na literatura método que possibilite a sua busca e identificação em menor tempo computacional.

Também, deve-se considerar o desenvolvimento de toda a rotina em linguagem de programação que a torne mais simples e rápida. Neste experimento, o desenvolvimento da aplicação não foi refinado pois não era este o foco do experimento.

Ainda, para que a rotina possa ser submetida a um experimento que possibilite a demonstração de resultados mais completos, é necessária a sua execução em instâncias maiores. Sendo portanto, no desenvolvimento de uma aplicação mais refinada, necessário considerar esta possibilidade.

Por fim, mesmo sendo o SCIP um resolvedor bastante eficiente, o seu tempo de processamento para a resolução de problemas em algum momento será inviável, considerando instâncias maiores ou com alguma característica específica. A demonstração deste pior caso e uma proposta de pré-processamento eficiente que viabilize a sua utilização é um interessante tema para trabalhos futuros.

REFERÊNCIAS

Appel, K., Haken, W. e Koch, J. (1977). Every planar map is four colorable. part ii: Reducibility. *Illinois Journal of Mathematics*, 21(3):491–567.

Beigel, R. e Eppstein, D. (2005). 3-coloring in time o (1.3289 n). *Journal of Algorithms*, 54(2):168–204.

Berlin, Z. I. (2021a). Scip - related work. `https://scipopt.org/index.php#work`. Acessado em 06/09/021.

Berlin, Z. I. (2021b). Scip - solving constraint integer programs. `https://scipopt.org/#scipoptsuite`. Acessado em 03/06/2021.

Berlin, Z. I. (2021c). Soplex: Overview. `https://soplex.zib.de/doc-5.0.2/html/`. Acessado em 24/06/2021.

Berthold, T., Gamrath, G., Gleixner, A., Heinz, S., Koch, T. e Shinano, Y. (2012). Solving mixed integer linear and nonlinear problems using the scip optimization suite. *ZIB-Report*.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM.

Burkardt, J. (2008). Cnf files. `https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html`. Acessado em 07/04/2022.

Caprara, A., Kroon, L., Monaci, M., Peeters, M. e Toth, P. (2007). Passenger railway optimization. *Handbooks in operations research and management science*, 14:129–187.

Carter, M. W., Price, C. C. e Rabadi, G. (2018). *Operations research: a practical introduction*. Chapman and Hall/CRC.

Casselgren, C. J. (2011). *On some graph coloring problems*. Tese de doutorado, Umeå universitet, Institutionen för matematik och matematisk statistik.

Chiarandini, M. e Gualandi, S. (2015). Bibliography on graph-vertex coloring. `https://imada.sdu.dk/~marco/gcp/`. Acessado em 07/07/2022.

Dantzig, G. B. e Wolfe, P. (1960). Decomposition principle for linear programs. *Operations research*, 8(1):101–111.

de Werra, D. (1985). An introduction to timetabling. *European journal of operational research*, 19(2):151–162.

Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K. et al. (2020). The scip optimization suite 7.0. *ZIB-Report*.

Gamst, A. (1986). Some lower bounds for a class of frequency assignment problems. *IEEE transactions on vehicular technology*, 35(1):8–14.

Garey, M. R. e Johnson, D. S. (1979). Computers and intractability. *A Guide to the*.

Garey, M. R., Johnson, D. S. e Stockmeyer, L. (1974). *Some simplified NP-complete problems*.

Graphviz (2022). Graphviz - documentation / output formats. `https://graphviz.org/docs/outputs/`. Acessado em 29/11/2022.

Gu, J. (1994). Optimization algorithms for the satisfiability (sat) problem. Em *Advances in Optimization and Approximation*, páginas 72–154. Springer.

Kubale, M. (2004). *Graph colorings*, volume 352. American Mathematical Soc.

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards*, 84(6):489.

Lima, C. L. G. d. (2016). *Um estudo sobre teoria dos grafos e o teorema das quatro cores*. Tese de doutorado, Universidade de São Paulo.

Luenberger, D. G., Ye, Y. et al. (1984). *Linear and nonlinear programming*, volume 2. Springer.

Matousek, J. e Gärtner, B. (2007). *Understanding and using linear programming*. Springer

Science & Business Media.

Mizuno, K. e Nishihara, S. (2008). Constructive generation of very hard 3-colorability instances. *Discrete Applied Mathematics*, 156(2):218–229.

Papadimitriou, C. H. e Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Ravindran, A. R. (2016). *Operations research and management science handbook*. Crc Press.

Skiena, S. (1991). *Implementing discrete mathematics: combinatorics and graph theory with*

Mathematica. Addison-Wesley Longman Publishing Co., Inc.

Sousa, L. (2001). O teorema das quatro cores. *Millenium*, páginas 125–151.

Weisstein, E. W. (2008). Vertex coloring. *https://mathworld. wolfram. com/*.

Woo, T.-K., Su, S. e Newman-Wolfe, R. (1991). Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm. *IEEE Transactions on Communications*, 39(12):1794–1801.